Origami with strings: protein folding by computer

Kevin Karplus
karplus@soe.ucsc.edu

Biomolecular Engineering Department
Undergraduate and Graduate Director, Bioinformatics
University of California, Santa Cruz
Outline of Talk

- What is Biomolecular Engineering? Bioinformatics?
- What is a protein?
- The folding problem and variants on it:
  - Local structure prediction
  - Fold recognition
  - Comparative modeling
  - “Ab initio” methods
  - Contact prediction
What is Biomolecular Engineering?

Engineering with, of, or for biomolecules. For example, with: using proteins as sensors or for self-assembly.

of: protein engineering—designing or artificially evolving proteins to have particular functions

for: designing high-throughput experimental methods to find out what molecules are present, how they are structured, and how they interact.
What is Bioinformatics?

Bioinformatics: using computers and statistics to make sense out of the mountains of data produced by high-throughput experiments.

- Genomics: finding important sequences in the genome and annotating them.
- Phylogenetics: “tree of life”.
- Systems biology: piecing together various control networks.
- DNA microarrays: what genes are turned on under what conditions.
- Proteomics: what proteins are present in a mixture.
- Protein structure prediction.
What is a protein?

There are many abstractions of a protein: a band on a gel, a string of letters, a mass spectrum, a set of 3D coordinates of atoms, a point in an interaction graph, . . . .

For us, a protein is a long skinny molecule (like a string of letter beads) that folds up consistently into a particular intricate shape.

The individual “beads” are amino acids, which have 6 atoms the same in each “bead” (the backbone atoms: N, H, CA, HA, C, O).

The final shape is different for different proteins and is essential to the function.

The protein shapes are important, but are expensive to determine experimentally.
Folding Problem

The Folding Problem:
If we are given a sequence of amino acids (the letters on a string of beads), can we predict how it folds up in 3-space?

```
MTMSRRNTDA ITIHSILDWI EDNLESPLSL EKVSERSGYS KWHLQRMFKK
ETGHSLGQYI RSRKMETIAQ KLKESNEPIL YLAERYGFES QQTLTRTFKN
YFDVPPHKYR MTNMQGESRF LHPLNHYS
```

Too hard!
The Fold-recognition Problem:
Given a sequence of amino acids $A$ (the target sequence) and a library of proteins with known 3-D structures (the template library), figure out which templates $A$ match best, and align the target to the templates.

- The backbone for the target sequence is predicted to be very similar to the backbone of the chosen template.
- Progress has been made on this problem, but we can usefully simplify further.
Remote-homology Problem

The Homology Problem:
Given a target sequence of amino acids and a library of protein sequences, figure out which sequences is similar to and align them to.

- No structure information is used, just sequence information. This makes the problem easier, but the results aren’t as good.

- This problem is fairly easy for recently diverged, very similar sequences, but difficult for more remote relationships.
New-fold prediction

What if there is *no* template we can use?

We can try to generate many conformations of the protein backbone and try to recognize the most protein-like of them.

Search space is huge, so we need a good conformation generator and a cheap cost function to evaluate conformations.
Hidden Markov Models

Hidden Markov Models (HMMs) are a very successful way to capture the variability possible in a family of proteins.

An HMM is a stochastic model—that is, it assigns a probability to every possible sequence.

An HMM is a finite-state machine with a probability for emitting each letter in each state, and with probabilities for making each transition between states.

Probabilities of letters sum to one for each state.

Probabilities of transitions out of each state sum to one for that state.

We also include null states that emit no letters, but have transition probabilities on their out-edges.
Circles are null states.

Squares are *match states*, each of which is paired with a null *delete state*. We call the match-delete pair a *fat state*.

Each fat state is visited exactly once on every path from Start to End.

Diamonds are *insert states*, and are used to represent possible extra amino acids that are not found in most of the sequences in the family being modeled.
What is single-track HMM looking for?

nostruct-align/3chy.t2k w0.5
Secondary structure Prediction

Instead of predicting the entire structure, we can predict local properties of the structure.

What local properties do we choose?

We want properties that are well-conserved through evolution, easily predicted, and useful for finding and aligning templates.

One popular choice is a 3-valued helix/strand/other alphabet—we have investigated many others. Typically, predictors get about 80% accuracy on 3-state prediction.

Many machine-learning methods have been applied to this problem, but the most successful is neural networks.
CASP Competition Experiment

- Everything published in literature “works”
- CASP set up as true blind test of prediction methods.
- Sequences of proteins about to be solved released to prediction community.
- Predictions registered with organizers.
- Experimental structures compared with solution by assessors.
- “Winners” get papers in *Proteins: Structure, Function, and Bioinformatics*. 
Predicting Local Structure

Want to predict some local property at each residue.

Local property can be emergent property of chain (such as being buried or being in a beta sheet).

Property should be conserved through evolution (at least as well as amino acid identity).

Property should be somewhat predictable (we gain information by predicting it).

Predicted property should aid in fold-recognition and alignment.

For ease of prediction and comparison, we look only at discrete properties (alphabets of properties).
Using Neural Net

We use neural nets to predict local properties.

Input is profile with probabilities of amino acids at each position of target chain, plus insertion and deletion probabilities.

Output is probability vector for local structure alphabet at each position.

Each layer takes as input windows of the chain in the previous layer and provides a probability vector in each position for its output.

We train neural net to maximize

$$\sum \log(P(\text{correct output}))$$.
Neural Net

Typical net has 4 layers and 6471 weight parameters:

<table>
<thead>
<tr>
<th>input/pos</th>
<th>window</th>
<th>output/pos</th>
<th>weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>5</td>
<td>15</td>
<td>1665</td>
</tr>
<tr>
<td>15</td>
<td>7</td>
<td>15</td>
<td>1590</td>
</tr>
<tr>
<td>15</td>
<td>9</td>
<td>15</td>
<td>2040</td>
</tr>
<tr>
<td>15</td>
<td>13</td>
<td>6</td>
<td>1176</td>
</tr>
</tbody>
</table>

Inputs

Hidden Layer 1

Hidden Layer 2

Hidden Layer 3

Output Layer
DSSP is a popular program to define secondary structure.

7-letter alphabet: EBGHSTL

- E = $\beta$ strand
- B = $\beta$ bridge
- G = $3_{10}$ helix
- H = $\alpha$ helix
- I = $\pi$ helix (very rare, so we lump in with H)
- S = bend
- T = turn
- L = everything else (DSSP uses space for L)
Yael Mandel-Gutfreund noticed that parallel and anti-parallel strands had different hydrophobicity patterns, implying that parallel/antiparallel can be predicted from sequence.

We created a new alphabet, splitting DSSP’s E into 6 letters:
HMMSTR $\phi$-$\psi$ alphabet

For HMMSTER, Bystroff did k-means classification of $\phi$-$\psi$ angle pairs into 10 classes (plus one class for cis peptides).

We used just the 10 classes, ignoring the $\omega$ angle.
Backbone geometry can be mostly summarized with one angle per residue:

We discretize into 11 classes:
de Brevern’s Protein Blocks

Clustered on 5-residue window of \( \phi-\psi \) angles:
Burial alphabets

Our second set of investigations was for a sampling of the many burial alphabets, which are discretizations of various accessibility or burial measures:

- solvent accessible surface area
- relative solvent accessible surface area
- neighborhood-count burial measures
Solvent Accessibility

- **Absolute SA**: area in square Ångstroms accessible to a water molecule, computed by DSSP.
- **Relative SA**: Absolute SA/ max SA for residue type (using Rost’s table for max SA).

![Graph showing frequency of occurrence vs. solvent accessibility](image)

**Frequency of occurrence**: $1e^{-05}$ to 0.1

**Solvent accessibility**: 17, 24, 46, 71, 106
Burial

- Define a sphere for each residue.
- Count the number of atoms or of residues within that sphere.
- Example: center = $C_\beta$, radius = 14 Å, count = $C_\beta$, quantize in 7 equi-probable bins.
Mutual Information

Mutual information between two random variables (letters of alphabet):

\[ MI(X, Y) = \sum_{i,j} P(i, j) \log \frac{P(i, j)}{P(i)P(j)} , \]

We look at mutual information between different alphabets at same position in protein. (redundancy)

We look at mutual information with one alphabet between corresponding positions on alignments of sequences.
Information Gain

Information gain is how much more we know about a variable after making a prediction.

\[ I(X) = \text{average} \log \frac{\hat{P}_i(X_i)}{P_0(X_i)} \]

- \( \hat{P}_i \) is predicted probability vector for position \( i \)
- \( X_i \) is actual observation at position \( i \)
- \( P_0 \) is background probability vector
## Conservation and Predictability

<table>
<thead>
<tr>
<th>Name</th>
<th>alphabet</th>
<th>MI with AA</th>
<th>conservation</th>
<th>predictability</th>
</tr>
</thead>
<tbody>
<tr>
<td>str</td>
<td>13</td>
<td>2.842</td>
<td>0.103</td>
<td>1.107</td>
</tr>
<tr>
<td>protein blocks</td>
<td>16</td>
<td>3.233</td>
<td>0.162</td>
<td>0.980</td>
</tr>
<tr>
<td>stride</td>
<td>6</td>
<td>2.182</td>
<td>0.088</td>
<td>0.904</td>
</tr>
<tr>
<td>DSSP</td>
<td>7</td>
<td>2.397</td>
<td>0.092</td>
<td>0.893</td>
</tr>
<tr>
<td>stride-EHL</td>
<td>3</td>
<td>1.546</td>
<td>0.075</td>
<td>0.861</td>
</tr>
<tr>
<td>DSSP-EHL</td>
<td>3</td>
<td>1.545</td>
<td>0.079</td>
<td>0.831</td>
</tr>
<tr>
<td>CB-16</td>
<td>7</td>
<td>2.783</td>
<td>0.089</td>
<td>0.682</td>
</tr>
<tr>
<td>CB-14</td>
<td>7</td>
<td>2.786</td>
<td>0.106</td>
<td>0.667</td>
</tr>
<tr>
<td>CB-12</td>
<td>7</td>
<td>2.769</td>
<td>0.124</td>
<td>0.640</td>
</tr>
<tr>
<td>rel SA</td>
<td>7</td>
<td>2.806</td>
<td>0.183</td>
<td>0.402</td>
</tr>
<tr>
<td>abs SA</td>
<td>7</td>
<td>2.804</td>
<td>0.250</td>
<td>0.382</td>
</tr>
</tbody>
</table>
Multi-track HMMs

We can also use alignments to build a two- or three-track target HMM:

- Amino-acid track (created from the multiple alignment).
- Local-structure track(s) with probabilities from neural net.
- Can align template (AA+local) to target model.
What is second track looking for?
Target-model Fold Recognition

- Find probable homologs of target sequence and make multiple alignment.
- Make secondary structure probability predictions based on multiple alignment.
- Build an HMM based on the multiple alignment and predicted 2ry structure (or just on multiple alignment).
- Score sequences and secondary structure sequences for proteins that have known structure (all sequences for AA-only, 8,000-11,000 representatives for multi-track).
- Select the best-scoring sequence(s) to use as templates.
Template-library Fold Recognition

- Build an HMM for each protein in the template library, based on the template sequence (and any homologs you can find).
- The HMM library has over 12,000 templates from PDB.
- For the fold-recognition problem, structure information can be used in building these models (though we currently don’t).
- Score target sequence with all models in the library.
- Select the best-scoring model(s) to use as templates.
Combined SAM-Txx method

Combine the costs from the template library search and the target library searches using different local structure alphabets.

Choose one of the many alignments of the target and template (whatever method gets best results in testing).

http://www.soe.ucsc.edu/research/compbio/SAM_T06/T06-query.html
Fold recognition results

Fold Recognition for 1415 SAM-T05 HMMs with w(?,amino-acid)=1

- average w(near-backbone-11)=0.6
- w(nsep)=0.1
- average w(near-backbone-11)=0.4
- w(nsep)=0.1
- w(str2)=0.25
- T2K w(str2)=0.2
- T2K aa-only

True positives / possible fold matches vs. False positives per query
Comparative modeling: T0232

RMSD = 5.158Å all-atom, 4.463Å $C_\alpha$
T0298 domain 2 (130–315)

RMSD= 2.468Å all-atom, 1.7567Å $C'_\alpha$, GDT=82.5%
best model 1 submitted to CASP7 (red=real)
Comparative modeling: T0348

RMSD = 11.8 Å $C_\alpha$, GDT = 58.2% (cartoon=real)
best model 1 by CASP7 GDT, Robetta1 slightly better.
Undertaker

Undertaker is UCSC’s attempt at a fragment-packing program.

Named because it optimizes burial.

Representation is 3D coordinates of all heavy atoms (not hydrogens).

Can replace backbone fragments (a la Rosetta) or full alignments—chain need not remain contiguous.

Conformations can borrow heavily from fold-recognition alignments, without having to lock in a particular alignment.

Use genetic algorithm with many conformation-change operators to do stochastic search.
Fragfinder

Fragments are provided to undertaker from 3 sources:

- **Generic fragments** (2-4 residues, exact sequence match) are obtained by reading in 500–1000 PDB files, and indexing all fragments.

- **Long specific fragments** (and full alignments) are obtained from the various target and template alignments generated during fold recognition.

- **Medium-length fragments** (9–12 residues long) for every position are generated from the HMMs with fragfinder, a new tool in the SAM suite.
Cost function

Cost function is modularly designed—easy to add or remove terms.

Cost function can include predictions of local properties by neural nets.

Clashes and hydrogen bonds are important components.

There are over 40 cost function components available: burial functions, disulfides, contact order, rotamer preference, radius of gyration, constraints, ...
We tried forcing various sheet topologies and selected 4 by hand.

Model 1 has right topology (5.912Å all-atom, 5.219Å $C_\alpha$).

Unconstrained cost function not good at choosing topology (two strands curled into helices).

Helices were too short.
Contact prediction

- Use mutual information between columns.
- Thin alignments aggressively (30%, 35%, 40%, 50%, 62%).
- Compute e-value for mutual info (correcting for small-sample effects).
- Compute rank of log(e-value) within protein.
- Feed log(e-values), log rank, contact potential, joint entropy, and separation along chain for pair, and amino-acid profile, predicted burial, and predicted secondary structure for each residue of pair into a neural net.
Evaluating contact prediction

Two measures of contact prediction:

Accuracy:

$$\frac{\sum \chi(i, j)}{\sum 1}$$

(favors short-range predictions, where contact probability is higher)

Weighted accuracy:

$$\frac{\sum \chi(i, j) \cdot \text{Prob}(\text{contact}|\text{separation} = |i - j|)}{\sum 1}$$

(1 if predictions no better than chance based on separation).
Contact prediction results

Accuracy of contact prediction, by protein

Weighted-accuracy of contact prediction, by protein
Target T0230 (FR/A)

⚠️ Good except for C-terminal loop and helix flopped wrong way.

⚠️ We have secondary structure right, including phase of beta strands.

⚠️ Contact prediction helped, but we put too much weight on it—decoys fit predictions better than real structure does.
Target T0230 (FR/A)
Target T0230 (FR/A)

Real structure with contact predictions:
Web sites

These slides:


SAM-T06 prediction server:

http://www.soe.ucsc.edu/research/compbio/SAM_T06/T06-query.html

CASP6 all our results and working notes:

http://www.soe.ucsc.edu/~karplus/casp6/

Predictions for all yeast proteins:

http://www.soe.ucsc.edu/~karplus/yeast/

UCSC bioinformatics (research and degree programs) info:

http://www.soe.ucsc.edu/research/compbio/

SAM tool suite info:

http://www.soe.ucsc.edu/research/compbio/sam.html